VARIABLE DEGREE SCHWARZ METHODS FOR THE IMPLICIT SOLUTION OF UNSTEADY COMPRESSIBLE NAVIER-STOKES EQUATIONS ON TWO-DIMENSIONAL UNSTRUCTURED MESHES*

Xiao-Chuan Cai†
Department of Computer Science
University of Colorado
Boulder, CO 80309
cai@cs.colorado.edu

Charbel Farhat
Department of Aerospace Engineering
University of Colorado
Boulder, CO 80309
charbel@alexandra.colorado.edu

Marcus Sarkis‡
Department of Computer Science
University of Colorado
Boulder, CO 80309
msarkis@cs.colorado.edu

Abstract

We report our experiences on using a new variant of the Schwarz preconditioned GMRES methods in the implicit solution of the unsteady compressible Navier-Stokes equations discretized on two-dimensional unstructured meshes. We first partition the global mesh with the recursive spectral bisection method into submeshes, and then we introduce a family of Schwarz methods, referred to as the Variable Degree Schwarz methods (VDS) on the overlapping submeshes. In VDS, the subdomain preconditioner is constructed by using a polynomial in two matrix variables, namely the matrix, in its *un-factorized* form, of the current time step k and another matrix, in its *factorized* form, obtained at a previous time step j. The degree of the matrix polynomial in each subdomain is determined automatically so that extra preconditioning is performed only in subdomains whose associated local matrices have large condition numbers. The extra preconditioning occurs often near the body of the airfoil. We show numerically that VDS is very effective. Unlike the well-known elliptic theory, we observe that the convergence rate of VDS preconditioned GMRES degenerates very mildly without a coarse space for reasonably large number of subdomains. We also study the effects of the overlapping size, the number of subdomains and the level of inexactness of the subdomain solvers. The other purpose of the study is to understand the robustness of the Schwarz methods with respect to flow parameters, such as the CFL, the free stream Mach number and the Reynolds number. Numerical results for both subsonic and transonic problems are reported.

* The work was supported in part by the NSF Grand Challenges Applications Group grant ASC-9217394 and the NASA HPCC Group grant NAG5-2218.

† The work was supported in part by the NSF grant ASC-9457534, and by NASA Contract No. NAS1-19480 while the author was in residence at the Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, VA 23681.

‡ The work was supported in part by the NSF grant ASC-9406582.

1. Introduction. The system of unsteady compressible Navier-Stokes (N.-S.) equations is a fundamental system in fluid dynamics. To be able to solve the system quickly and accurately in complex geometry is one of the ultimate goals of computational fluid dynamics [20]. To achieve the goal, several important techniques have been developed in the past few years, such as unstructured grid generations for complex geometry, stable, conservative discretizations of the N.-S. equations, unstructured grid partitionings, as well as the powerful, robust implicit preconditioned iterative solvers discussed below. Most of the techniques are developed for steady state calculations, see e.g., [1, 5, 16] and references therein. In this paper, we put all the techniques together and introduce a robust domain decomposition based fast preconditioned iterative method for the time accurate solution of unsteady problems.

We study implicit methods for solving unsteady N.-S. equations discretized on two-dimensional unstructured meshes with a combined finite element/finite volume scheme for the spatial variables and a simple backward Euler scheme for the temporal variable. It is well-known that the main advantage of implicit methods is that they allow the time steps to be determined solely based on the physics of the fluid flow, not on the stability property of the time discretization scheme, [4, 30, 31]. However, to advance in time, a large, sparse, nonsymmetric linear system of equations must be constructed and solved at each time step. Depending of the size of the time step, and several other flow parameters, the conditioning of the matrix may change from well-conditioned to mildly ill-conditioned. And due to the complexity of the flow pattern, at a given time step, the matrix may be ill-conditioned in certain subregions near the airfoil and relatively well-conditioned elsewhere. Details about the local conditioning of the matrix will be discussed later. To solve these systems iteratively, it is necessary to have a family of preconditioners whose strength can be adjusted locally in each subdomain according to the flow condition. Overlapping Schwarz methods (OSM) is a family of preconditioners for solving large sparse linear systems arising from the discretization of partial differential equations, see e.g. [6, 8, 11, 27]. It was originally designed for scalar linear elliptic problems. We find OSM to be very attractive for our purpose because (1) they are more parallelizable than the popularly used global ILU preconditioners; (2) they are efficient for nonsymmetric and indefinite problems; (3) they have mesh independent convergence rates, at least for elliptic finite element problems; (4) they have adjustable strength controlled by using the inexact solution techniques for solving local problems. We shall further explore the flexibility of OSM at the subdomain level and introduce a new variant below.

It is well-known that when constructing a preconditioner for solving a single system of linear equations, $Au = f$, all the information needs to be from the matrix A. However, the issue for time dependent problems is different. A sequence of inter-related systems $A^{(k)}u = f^{(k)}$ has to be solved. If the matrix, and especially in its (often inexactly) factorized form, obtained at a previous time step can be properly used, then the preconditioner at the current time step can be obtained cheaply. More precisely speaking, at each time step, we solve the global linear system by a preconditioned GMRES method ([26]) and in the preconditioning stage, following the general overlapping Schwarz framework, we solve the local subdomain problems by another preconditioned GMRES method, with different preconditioners and stopping conditions. In each subdomain the preconditioner is built by using a polynomial in two matrix variables, namely the matrix, in its *un-factorized* form, of the current time step k and another matrix, in its *factorized* form, obtained at a previous time step j. The degree of the matrix polynomial reflects the conditioning of the subdomain matrix. Note that classical Schwarz methods correspond to the case where the degree of

the matrix polynomials always equals to one. In our method, the degree of the polynomial varies from subdomain to subdomain depending the flow conditions, and therefore we refer to the methods as variable degree Schwarz methods (VDS).

In this paper, we also investigate the difference between the Schwarz family of preconditioners and other methods such as the simple Jacobi iterative method and the global ILU preconditioned iterative method. Within the Schwarz preconditioners, we try to understand the role of the overlapping size between subdomains, the effect of the number of subdomains, and the effect of the inexact subdomain solvers. Since the construction of the preconditioner is expensive, we explore the possibility of re-using the preconditioner for several time steps.

We restrict our attention to sequential computers, and single level Schwarz algorithms. For steady state problems, some studies can be found in [18]. For other recent developments in unsteady calculations, we refer the readers to [2, 30, 31]. Our focus is on the Schwarz algorithms, therefore only the simplest time discretization, namely the first order backward Euler scheme, is considered in this paper. Higher order schemes and their influence on the Schwarz preconditioners will be discussed in a forthcoming paper. The physical model we choose to test our algorithms contains a single element NACA0012 airfoil at a rather large angle of attack with a modest Reynolds number. Both subsonic and transonic cases are studied in the paper. The paper is organized as follows. In §2, we discuss the unsteady compressible N.-S. equations in the conservative form, the boundary conditions and a discretization scheme. In §3, we study a preconditioned iterative method and introduce a new variant of the overlapping Schwarz preconditioners. Numerical experiments for both subsonic and transonic flows are reported in §4. §5 includes a few final remarks.

2. The two-dimensional unsteady N.-S. equations. In this section, we describe the two-dimensional unsteady compressible N.-S. equations in its conservative form. We also discuss the spatial and temporal discretizations of the equations on unstructured meshes. Following the notions of Farhat, Fezoui and Lanteri [12, 13], and Fezoui and Stoufflet [15], for the spatial variables, we use a combined finite element/finite volume scheme and for the temporal variable we use a simple backward Euler method. The scheme is of second order in space and first order in time.

2.1. Governing equations. Let $\Omega \subset \Re^2$ be the flow domain and Γ its boundary, as shown in Fig. 1. The conservative form of the N.-S. equations is given by

$$\frac{\partial W}{\partial t} + \nabla_{\vec{x}} \cdot \mathcal{F}(W(\vec{x},t)) = \frac{1}{Re} \nabla_{\vec{x}} \cdot \mathcal{R}(W(\vec{x},t)), \tag{1}$$

where $\vec{x}$ and t denote the spatial and temporal variables, and

$$W = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ E \end{pmatrix}, \quad \nabla_{\vec{x}} = \begin{pmatrix} \frac{\partial}{\partial x} \\ \\ \frac{\partial}{\partial y} \end{pmatrix}, \quad \mathcal{F}(W) = \begin{pmatrix} F_1(W) \\ F_2(W) \end{pmatrix}, \quad \mathcal{R}(W) = \begin{pmatrix} R_1(W) \\ R_2(W) \end{pmatrix}.$$

Here the functions F_1 and F_2 denote the convective fluxes

$$F_1(W) = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ v(E+p) \end{pmatrix}, \qquad F_2(W) = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(E+p) \end{pmatrix}$$

and the functions R_1 and R_2 denote the diffusive fluxes

$$R_1(W) = \begin{pmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ u\tau_{xx} + v\tau_{xy} + \frac{\gamma k}{Pr}\frac{\delta \varepsilon}{\delta x} \end{pmatrix}, \quad R_2(W) = \begin{pmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ \tau_{xy} + v\tau_{yy} + \frac{\gamma k}{Pr}\frac{\delta \varepsilon}{\delta y} \end{pmatrix}.$$

In the above expressions, ρ is the density, $\vec{U} = (u, v)^T$ is the velocity vector, E is the total energy per unit of volume, and p is the pressure. These variables are related by the state equation for perfect gas

$$p = (\gamma - 1)\left(E - \frac{1}{2}\rho\|\vec{U}\|^2\right),$$

where γ denotes the ratio of specific heats ($\gamma = 1.4$, for air). The specific internal energy ε is related to the temperature via

$$\varepsilon = c_v T = \frac{E}{\rho} - \frac{1}{2}\|\vec{U}\|^2.$$

In the diffusive fluxes, τ_{xx}, τ_{xy}, and τ_{yy} are the components of the two-dimensional Cauchy stress tensor, k is the normalized thermal conductivity, $Pr = \mu_0 c_p / k_0$ is the Prandtl number ($Pr = 0.72$, for air), and $Re = \rho_0 \vec{U}_0 L_0 / \mu_0$ is the Reynolds number, where ρ_0, $\vec{U}_0$, L_0, and μ_0 denote the characteristic density, velocity, length, and diffusivity, respectively. The components of Cauchy stress tensor are related to the velocity via

$$\tau_{xx} = \frac{2}{3}\mu\left(2\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y}\right), \quad \tau_{yy} = \frac{2}{3}\mu\left(2\frac{\partial v}{\partial y} - \frac{\partial u}{\partial x}\right), \quad \tau_{xy} = \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right),$$

where μ denotes the normalized viscosity.

2.2. Boundary conditions. We are interested in unsteady, external flows around an airfoil as pictured in Fig.1. The domain boundary is $\Gamma = \Gamma_w \cup \Gamma_\infty$ and the far field velocity is $\vec{U}_\infty$. On the wall boundary Γ_w, a no-slip condition on $\vec{U}$ and a Dirichlet condition on the temperature T are imposed, i.e.,

$$\vec{U} = \vec{0}, \quad \text{and} \quad T = T_w. \tag{2}$$

No boundary condition is specified for the density. In the far field, the viscous effect is assumed to be negligible, therefore a uniform free-stream velocity $\vec{U}_\infty$ is imposed on Γ_∞

$$\rho = 1, \quad \vec{U}_\infty = \begin{pmatrix} cos\ \alpha \\ sin\ \alpha \end{pmatrix}, \quad \text{and} \quad p_\infty = \frac{1}{\gamma M_\infty^2}, \tag{3}$$

where α is the angle of attack, and M_∞ is the free-stream Mach number.

2.3. Discretization. Let the temporal variable t be discretized as $t^k = t^{k-1} + \delta t^k$, where δt^k is the discrete time increment and $t^0 = 0$. We consider the increment $\delta W^k = W^k - W^{k-1}$, where W^k is an approximation of $W(\cdot, t^k)$. Note that when an algorithm is written in the "delta" form [3, 28], the increment δW^k is the unknown variable rather than W^k. Here, we use a first-order finite difference approximation for the temporal variable, namely, the backward Euler scheme given as

$$\frac{\delta W^k}{\delta t^k} + (\nabla_W(\mathcal{F}^{k-1}) \cdot \nabla_{\vec{x}})\,\delta W^k - \frac{1}{Re}(\nabla_W(\mathcal{R}^{k-1}) \cdot \nabla_{\vec{x}})\,\delta W^k \tag{4}$$

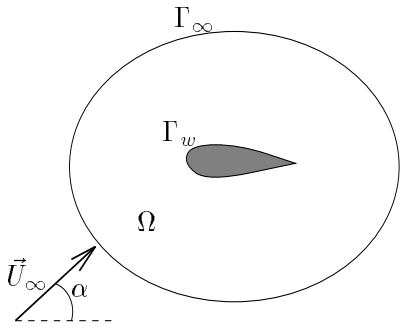

FIG. 1. *The computational domain Ω and its wall and far field boundaries.*

$$= \nabla \cdot (-\mathcal{F}^{k-1}) + \frac{1}{Re} \nabla \cdot \mathcal{R}^{k-1},$$

where $\mathcal{F}^{k-1}$ and $\mathcal{R}^{k-1}$ are approximations of $\mathcal{F}(W(\cdot, t^{k-1}))$ and $\mathcal{R}(W(\cdot, t^{k-1}))$, respectively. We determine the time step size in the following way. Let CFL be a pre-selected positive number. For each element, with size h_i, of the finite element mesh, we define an element time step size by

$$\delta t_i^k = h_i \frac{\text{CFL}}{(C_i + \|U_i\|_2) + 2/(Re\, Pr\, h_i)}$$

and then the global time step is defined by

$$\delta t^k = \min_i \{\delta t_i^k\}. \tag{5}$$

Here C_i is the element sound speed, and U_i is the element velocity vector.

The computational domain is discretized by a triangular grid as pictured in Fig.2. We use unstructured grids since they provide flexibility for tessellating about complex geometries and for adapting to flow features, such as shocks and boundary layers. We locate the variables at the vertices of the grid, which gives rise to a cell-vertex scheme. The space of solutions is taken to be the space of piecewise linear continuous functions. The discrete system is obtained via a mixed Galerkin finite element/finite volume formulation; see Farhat et al. [12, 13], and Fezoui and Stoufflet [15] for details. In short, the discretized system for (4) is obtained as follows:

- For the time derivative of (4) we use a "mass-lumping" technique, in which we replace the mass matrix by some diagonal matrix.
- For the convective terms of the left-hand side of (4), we use a first order scheme that is an extension of Van Leer's MUSCL [29] scheme to the case of unstructured grids (see Fezoui and Stoufflet [15]) with a Roe approximate Riemann solver [24].
- For the diffusive terms of the left-hand side of (4), we use a Galerkin finite element (first-order quadrature integration).
- For the convective terms of the right-hand side of (4), we use a second-order scheme that is again an extension of Van Leer's MUSCL scheme to the case of unstructured grids (see Fezoui and Stoufflet [15]) with a Roe approximate Riemann solver [24]. We also use Van Albada's limiting procedure to reduce numerical oscillations of the solutions.
- For the diffusive terms of the right-hand side of (4), we use a regular Galerkin finite element method (second-order quadrature integration).

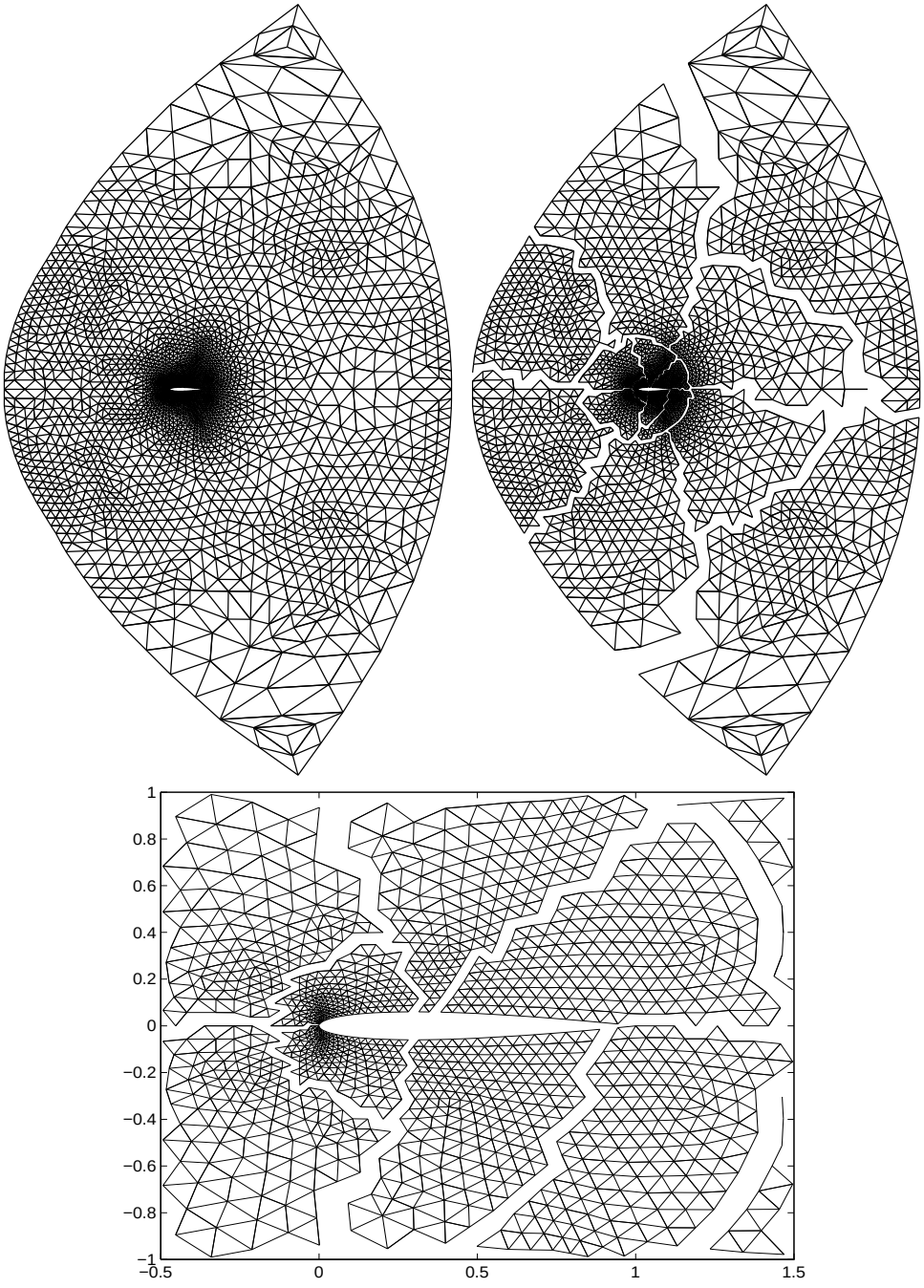

FIG. 2. *The top left figure shows the un-decomposed nonuniform shape regular finite element mesh. The top right figure shows the decomposition of the mesh into non-overlapping subdomains, and the bottom figure is a blow-up of the figure around the airfoil.*

Although both approximations we use for the left-hand side of (4) are spatially first-order, they operate on the increment δW^k and as a consequence the resulting scheme is spatially second-order for any fixed δt. We assume W^k satisfies strongly the wall boundary conditions on Γ_w and the far field boundary conditions at infinity. For the initial values, W^0 satisfies strongly the wall boundary conditions on Γ_w and the far field boundary conditions at Γ_∞. For the initial values and at the interior nodes of Ω, W^0 takes the free-stream boundary condition.

Putting pieces of the discretization together, at each time step, we obtain a large, sparse, generally nonsymmetric linear system of equations of the form

$$\left(\frac{D^{(k)}}{\delta t^k} + B^{(k)}\right) u^k = f^k, \tag{6}$$

where $D^{(k)}$ is a diagonal, lumped mass matrix, $B^{(k)}$ is the sum of the convective and diffusive terms on the left-hand side of (4) discretized by the finite volume and finite element methods, respectively, f^k is the discretized right-hand side of (4), and u^k is the approximate nodal value of δW^k at the finite element mesh points. We solve (6) by a preconditioned Krylov iterative method with a preconditioner $M^{(k)}$ to a certain *linear tolerance* τ, i.e.,

$$\left\| M^{(k)} \left[\left(\frac{D^{(k)}}{\delta t^k} + B^{(k)}\right) u^k - f^k \right] \right\|_2 \leq \tau \| M^{(k)} f^k \|_2. \tag{7}$$

To simplify the discussion, we shall use $A^{(k)} = \frac{D^{(k)}}{\delta t^k} + B^{(k)}$ in the rest of the paper. The preconditioner $M^{(k)}$ will be introduced in the next section.

3. Solution methods and variable degree Schwarz preconditioners. In this section, we first briefly recall the classical additive and multiplicative Schwarz algorithms. Then we introduce a variable degree version of Schwarz algorithms that is more suitable for solving time dependent problems. Unlike steady state problems, if certain information or matrix factorization obtained at the previous time steps can be used a great deal of calculations can be saved.

Let us review the Schwarz algorithms. Suppose that, at time step k, we need to solve a linear system of equations

$$A^{(k)} u^k = f^k,$$

where $A^{(k)}$ is an explicitly constructed, nonsymmetric and sparse matrix with symmetric non-zero pattern. Since our main interest is on the multicomponent N.-S. equations in two-dimensional spaces, each element of $A^{(k)}$ can be considered as a small 4×4 matrix, and each unknown of the vector $u^{(k)}$ is a '4-size' vector. Thus, there is a bijection between unknowns and vertices. We denote the set of vertices (or nodes) by $\mathcal{N} = \{1, \cdots, n\}$, where n represents the total number of nodes (or unknowns). To define algebraic Schwarz algorithms, see e.g. [6], we first partition the set $\mathcal{N}$ into n_0 nonoverlapping subsets $\{\mathcal{N}_i\}$ whose union is $\mathcal{N}$. We use the TOP/DOMDEC mesh partitioning package of Farhat et al. [14] to obtain sets $\mathcal{N}_i$. We use the recursive spectral bisection method ([22]) with certain optimization to obtain roughly the same number of interior and boundary nodes in each $\mathcal{N}_i$, and also to obtain good aspect ratio on the subgrids. To generate an overlapping partitioning with overlap

ovlp, we further expand each subgrid $\mathcal{N}_i$ by *ovlp* number of neighboring nodes, denoted as $\tilde{\mathcal{N}}_i$.

We denote by L_i the vector space spanned by the set $\tilde{\mathcal{N}}_i$. For each subspace L_i, we define an orthogonal projection operator I_i as follows: I_i is a $n \times n$ block diagonal matrix whose elements are 4×4 identity matrices if the corresponding nodes belong to $\tilde{\mathcal{N}}_i$ and to 4×4 zero matrices otherwise. With this we define

$$A_i^{(k)} = I_i A^{(k)} I_i \ ,$$

which is an extension to the whole subspace, of the restriction of $A^{(k)}$ to L_i. Note that although $A_i^{(k)}$ is not invertible in the full space, its restriction to the subspace spanned by $\tilde{\mathcal{N}}_i$ is nonsingular, and we define its inverse by

$$(A_i^{(k)})^{-1} \equiv I_i \left((A_i^{(k)})_{|L_i} \right)^{-1} I_i.$$

The classical additive and multiplicative Schwarz algorithms can now be simply described as follows: Solve the linear system

$$M A^{(k)} u = M f^{(k)}$$

by a Krylov subspace method, where

$$M = (A_i^{(k)})^{-1} + \cdots + (A_{n_0}^{(k)})^{-1} \ , \tag{8}$$

for the additive Schwarz algorithm, and

$$M\, A^{(k)} = I - \left(I - (A_i^{(k)})^{-1} A^{(k)} \right) \cdots \left(I - (A_{n_0}^{(k)})^{-1} A^{(k)} \right) \tag{9}$$

for the multiplicative Schwarz algorithm.

It has been shown that the above mentioned algorithms are very successful for steady state CFD problems, see e.g., [5, 21] and also [9]. There are three major steps in the construction of the Schwarz preconditioners, namely 1) the construction of the matrix $A^{(k)}$; 2) the construction of the matrices $A_i^{(k)}$; and 3) the incomplete factorization of the matrices $A_i^{(k)}$. In fact Step 1) is not necessary since the matrices constructed in Step 2) can be used to calculate the matrix-vector multiplications. Since we are interested in implicit methods, Step 2) has to be done at every time step no matter how expensive it is. One expensive step in the construction of the preconditions as formulated above for time dependent problems is Step 3). One way to avoid the frequent factorization of $A_i^{(k)}$ is to simply use some old factorized matrix $A_i^{(j)}$ calculated at time step j, where $j < k$. However, this method may not be very effective if j and k are too far apart. More discussions on using frozen preconditioners can be found later in the paper.

Another problem with the Schwarz preconditioners (8) and (9) is that all subdomains are treated equally in terms of the level of preconditioning in the sense that the number of applications of $(A_i^{(k)})^{-1}$, or its inexact version, is the same on all subdomains, regardless of the fact that the subdomain matrices $A_i^{(k)}$ have vary different condition numbers. Physically speaking, the behavior of the partial differential equations in subdomains near the body of the airfoil, or near the shocks is very different from the regions that are far from the subdomains where the real actions take place.

Here we propose a method that places different level of preconditioning in different subdomains and will also show by numerical experiments that the methods remain effective even if j and k are far apart from each other. The idea is simple. We replace the required matrix-vector multiply in (8) or (9)

$$w = (A_i^{(k)})^{-1} v \tag{10}$$

by another *iterative procedure* with $(B_i^{(j)})^{-1}$ as the preconditioner. Here $B_i^{(j)}$ is an incomplete factorization of $A_i^{(j)}$ with certain levels of fill-ins. More precisely speaking, to obtain w for a given v, we run several steps of GMRES in the subspace L_i to drive the residual

$$\left\| (B_i^{(j)})^{-1}(v - A_i^{(k)}\tilde{w}) \right\|_2 \leq \delta \left\| (B_i^{(j)})^{-1} v \right\|_2 . \tag{11}$$

We then set $w := \tilde{w}$. Here δ is pre-selected small value. Examples can be found in §4 of this paper. In the matrix language, we replace the matrix $(A_i^{(k)})^{-1}$ in (10) by a matrix polynomial

$$poly_i \left((B_i^{(j)})^{-1} A_i^{(k)} \right)$$

of a certain degree, which depends of the number of GMRES iterations needed in the subspace L_i. To put them into a single form, the additive Schwarz preconditioner becomes

$$M = poly_1 \left((B_1^{(j)})^{-1} A_1^{(k)} \right) + \cdots + poly_{n_0} \left((B_{n_0}^{(j)})^{-1} A_{n_0}^{(k)} \right) .$$

Note that this preconditioner does not contain $(A_i^{(k)})^{-1}$, but it contains certain spectral information of $(A_i^{(k)})^{-1}$. This makes it very effective. In fact, M is a truncated series representation of $(A_i^{(k)})^{-1}$ based on a splitting of $A_i^{(k)}$ into the sum of $B_i^{(j)}$ and $A_i^{(k)} - B_i^{(j)}$. A discussion on a related polynomial preconditioning method can be found in [17]. We note that in a given subdomain, the number of GMRES iterations, or the degree of the polynomial, is determined by the conditioning of the local stiffness matrix. The multiplicative version can be constructed in a similar way. The extension to the local multiplicative Schwarz method ([7]) is also straightforward.

We remark that since the preconditioner changes in the GMRES loop due to the stopping condition determined by δ, it is generally more appropriate to use the so-called flexible GMRES [25], which is slightly more expensive than the regular one. We do not use the flexible GMRES since the regular GMRES presents no problem for our test cases. The implementation of the methods is rather complicated because of the use of multi-layered Krylov iterations as a global, or outer, and also local solvers. PETSc makes our numerical tests possible. More details regarding to the implementation will be discussed in the next section.

4. Numerical results. The goal of this section is to demonstrate the usefulness of the family of VDS preconditioners in the implicit solution of compressible flow problems, and to compare the effectiveness of the methods with various other methods, such as the pointwise Jacobi iterative method and the global ILU(0) preconditioned GMRES method, for both subsonic and transonic flows. The experiments were performed on a DEC Alpha workstation (275MHz, 512MB memory), and the software was written by using the newly developed package PETSc [19] of the Argonne National Laboratory. All arithmetic operations are in

double precision. The system BLAS library (dxml [10]) was used. Only sequential results are reported here. A parallel version of the code is being developed, and the results will be reported in the future. Here we apply our computational algorithms to the simulation of two-dimensional low Reynolds number chaotic flows past a NACA0012 airfoil at high angle of attack and two different Mach numbers. It was shown in Pulliam [23] that such flows can be resolved with a reasonable number of grid points. The accuracy of the computed solutions are substantiated by successive mesh refinements and comparisons with the results were reported in [23]. No steady state solutions exists for both test cases described below.

Test 1: The subsonic case with free stream Mach number $M_\infty = 0.1$ and $Re = 800.0$. We use a pre-generated shape regular triangular mesh with 12280 nodes; see Fig.2 for example. The Mach surfaces of the computed solution at various time steps are given in Fig.5.

Test 2: The transonic case with free stream Mach number $M_\infty = 0.84$ and $Re = 1600.0$. We use a mesh with 48792 nodes obtained by uniformly refining the mesh used in **Test 1**. The Mach surfaces of the computed solution at various time steps are given in Fig.6.

TABLE 1

Total CPU hours and time steps spent for calculating the lift curves using explicit and implicit methods with different CFL numbers. The total non-dimensionalized time interval is (0, 25) for the $M_\infty = 0.1$ case and (0, 10) for the $M_\infty = 0.84$ case.

		Expl. CFL=0.8	Impl. CFL=25	Impl. CFL=50	Impl. CFL=100
$M_\infty = 0.1$	CPU(hours)	54.75	22.84	13.26	7.58
Mesh12k	Time steps	196091	6018	3009	1504
$M_\infty = 0.84$	CPU(hours)	51.89	14.26	9.20	6.06
Mesh48k	Time steps	48216	1389	694	347

In the following discussions, we shall refer to these two meshes as "Mesh12k" and "Mesh48k", respectively. In the implementation of Schwarz preconditioners, we partition the mesh by using the TOP/DOMDEC package [14], which implements the recursive spectral bisection method. We require that all subdomains have more or less the same number of mesh points. An effort is made to reduce the number of mesh points along the interfaces of subdomains, which may be needed later in our parallel code to reduce the communication cost. The mesh generation and partitioning are considered as pre-processing steps, and therefore not counted toward the CPU time reported in Table 1. The sparse matrix defined by (4) is constructed at every time step, and stored in the Compressed Sparse Row format. The subdomain matrices are obtained by taking elements, according to a pre-selected index set, from the global matrix. A symbolic ILU(0) factorization of the subdomain matrix is performed at the very first time step, and re-used at all the later time steps. This is possible due to the fact that the matrices, constructed at every time step, share the same non-zero pattern. We also tested the ILU(k) ($k > 0$) preconditioners, which are not competitive with ILU(0) in terms of the CPU time in our implementation for both test cases. We remark that if ILU with drop tolerance is used then the non-zero pattern of the matrices may change and therefore the symbolic factorization may not be very useful.

We note that at the beginning of the flow movement, i.e., when the non-dimensionalized time $t \leq 1.0$, the flow changes so drastically that the use of any δt^n that makes the corresponding CFL number larger than 1.0 would result in the loss of time accuracy for the

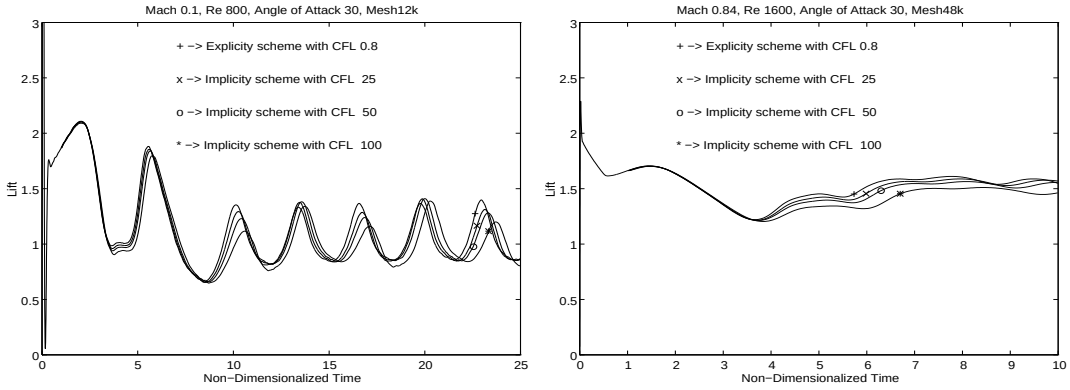

FIG. 3. *Lift curves obtained by explicit and implicit methods. The left figure shows the subsonic cases with* $M_\infty = 0.1$, $Re = 800.0$ *and the number of mesh points is* 12280. *The right figure is for the transonic cases with* $M_\infty = 0.84$, $Re = 1600.0$ *and the number of mesh points is* 48792.

entire calculation. This implies that small δt^n have to be used when $t \leq 1.0$, and therefore, the implicit method has to be abandoned for this initial period of time. In our experiments, the implicit solver is turned on at $t = 1.0$. The solution for the period $0 < t \leq 1.0$ is obtained with the explicit method with CFL=0.8.

4.1. The effect of using large CFL numbers. One of the biggest advantages of implicit methods is that large time steps, or large CFL numbers, can be used. We first examine this claim by comparing the time accuracy of two solutions obtained by using the implicit methods with CFL=25, 50 and 100, respectively, and the solution obtained by a second order (in time and space) explicit method, [12]. From Fig.3, one can easily see that no two different CFL numbers give identical solutions. However, the important thing for engineering purposes is to capture correctly the "phase" and "amplitude" of the lifts. Small errors in amplitude and phases are usually admissible. This can often substantially shorten the turnaround times in the initial aerodynamic design and analysis.

For the implicit methods, we use the multiplicative VDS preconditioned GMRES method as the global linear solver. The subdomain preconditioners are re-computed at every 5 time steps. In the Schwarz methods, we use 8 subdomains, and ovlp = 1. Each subdomain problem is solved by an ILU(0) preconditioned GMRES method with the stopping tolerance set to $\delta = 10^{-1}$. The stopping tolerance for the global linear system solver is $\tau = 10^{-3}$. We also test several cases with smaller τ, such as 10^{-4} and 10^{-5}. The resulting lift curves are not distinguishable from the ones shown in Fig.3.

In Table 1, we report the total number of time steps and CPU time in hours spent on the entire computation, not including the mesh generation and partitioning. We observe from our experiments that even with a CFL number, 100, not much time accuracy is lost for a certain period of time, see Fig.3. However, if the time accuracy is required for a longer period of time, we do recommend a smaller CFL number. We remark that our discussions here on the effects on using large CFL numbers is based on our first order time discretization scheme. The results may change slightly if higher order schemes are used.

4.2. The Schwarz parameters. The number of subdomains and the size of overlap are two important parameters for Schwarz methods. We here test the additive and multiplicative VDS methods for both test cases. Instead of running the entire calculation as

we did in the previous section, we run the tests for only 100 time steps starting at $t = 1.0$. In terms of the non-dimensional time, this means time intervals $(1, 1.86)$ for **Test 1** and $(1, 2.28)$ for **Test 2**.

In the rest of the paper, we shall use **MaxIt** to denote the maximum number of global GMRES iterations and **TotalIt** the total numbers of global GMRES iterations within this 100 linear system solves. To measure the approximate cost of the methods without including any machine dependent factors, we use **EMatVec** to denote the equivalent number of matrix-vector multiplications, which includes the actual stiffness matrix-vector multiplications and the preconditioning-matrix-vector multiplications. Since different subdomains may need different number of matrix-vector multiplications, we take the average over all subdomains and convert it into a multiple of the equivalent global matrix-vector multiplications. Suppose that $\tilde{n}$ is the size of the global matrix. Note that $\tilde{n}$ is 4 times the number of mesh points. Then, one global matrix-vector multiplication requires roughly $28\tilde{n}$ flops. Our primary iterative solver GMRES has a complexity of $I(I+2)\tilde{n} + I\times$(cost of a preconditioned matrix-vector multiplication). Here I is the number of iterations. For example, the pure GMRES cost, without counting the cost of the matrix-vector multiplications, for 4 iterations is about $24\tilde{n}$, which is a little less than the cost of doing 1 global stiffness matrix-vector multiplication.

Let us first discuss the dependence of the convergence rate on the number of subdomains. We use 5 different decompositions of Ω, with both Mesh12k and Mesh48k. The number of subdomains goes from 8 to 128. We run both **Test 1** and **2**, with ovlp equals to one fine mesh cell. In Table 2, we present the maximum number of global GMRES iterations within one hundred time steps and its corresponding **EMatVec**. If multiplicative VDS is used even without the special subdomain coloring or ordering, **MaxIt** is independent of the number of subdomains, for reasonably large number of subdomains, such as 128. An interesting case is shown on the top left portion of Table 2, which indicates that if additive VDS is used for the subsonic problem, the number of maximum iterations does grow, though not very fast, as the number of subdomains becomes large. In this case, we believe that a coarse space may be useful to reduce the dependence on the number of subdomains. However, we have not implemented the coarse grid solver yet. For transonic problems, our tests show that the use of a coarse level grid is not necessary with both additive and multiplicative VDS preconditioners.

Whether overlap is useful or not is a rather subtle issue. It depends on the global linear stopping parameter τ defined in (7) and the local linear stopping parameter δ defined in (11). In Table 3, we report the case for $\delta = 10^{-6}$ and varying τ. According to the results in Table 3 and a large number of other tests we did (not being reported here), large overlaps can reduce the number of iterations and CPU time only if the stopping parameter δ is small. In our situation when $\tau = 10^{-3}$, we find $\delta = 10^{-1}$ offers the best CPU results, and therefore we do not need large overlaps. In the rest of the tests, we shall use this set of τ and δ, with 1 overlap.

We next exam the VDS methods. We focus on the case with 8 subdomains, and use GMRES/ILU(0) as the local subdomain solvers. The partitionings used for Mesh12k and Mesh48k are different. The subdomains are numbered as in Fig.4. The results obtained for one hundreds time steps starting at $t = 1.0$ are summerized in Table 4. We have also run the test for t equals to other values and the results are more or less the same. We use the same local stopping condition, namely $\delta = 10^{-1}$ for all subdomains and for both subsonic and transonic problems. It turns out the required degrees of local preconditioning

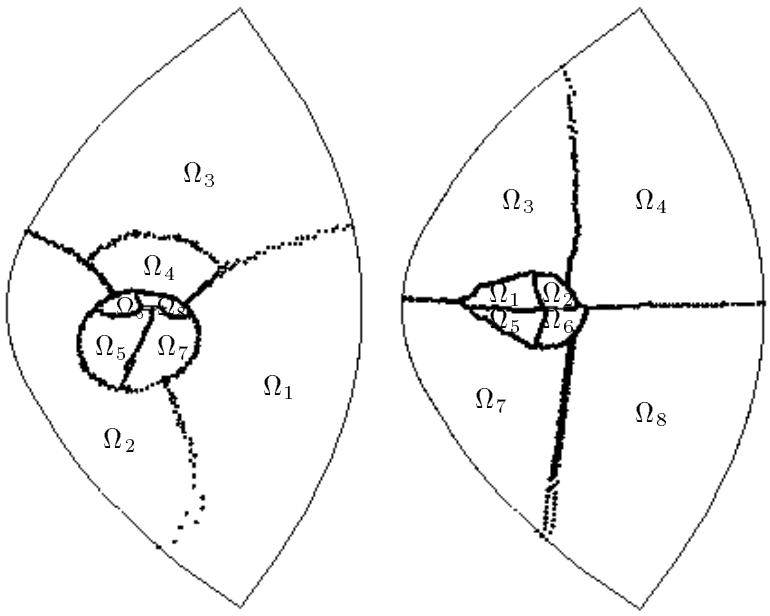

FIG. 4. *The left figure shows the partitioning of Mesh12k into 8 subdomains and the right ones shows that for Mesh48k.*

polynomials are quite different. For the subsonic case, subdomains Ω_6 and Ω_8 need more iterations (4 and 6 respectively) than other subdomains. The left picture of Fig.4 shows that these two subdomains cover the top portion of the airfoil. Only two iterations are needed for subdomains that are far away from the airfoil, such as Ω_1, Ω_2 and Ω_3. The number of iterations reflects the conditioning of the subdomain matrix. For the transonic case, all subdomains need either one or two more iterations.

For both test problems, Table 2 and Table 4 show that both the number of global iterations and the number of local iterations are surprisingly small, which indicate that the linear system of equations (6) is in fact not too ill-conditioned. We believe that this is due to the use of relatively small time steps (5).

4.3. A comparison with the pointwise Jacobi iterative method. For comparison purpose, we solve both test problems by using the simplest iterative method, namely the pointwise Jacobi method, which is often referred to as the Jacobi preconditioned Richardson's method. Jacobi method has a few very attractive features, such as being easy to parallelize. Note that for multicomponent test problems, a point corresponds to a 4×4 matrix. When using the Jacobi method, a dense 4×4 matrix needs to be inverted at every mesh point. In our experiments, at each time step before solving the linear system, we compute the inverse of these 4×4 matrices explicitly and save them for the Jacobian iterations. As before, we run the tests for one hundred time steps and record the maximum number of iterations as well as the total number of iterations, see Table 5. We observe that, in terms of iteration numbers for solving linear systems, the transonic problem is easier to handle than the subsonic problem, which is more of an elliptic system. Comparing the TotalIt, which equals to the total number of EMatVec, in Table 5, and the total EMatVec numbers in Table 2, we found that Jacobi is considerably more expensive than the Schwarz preconditioned GMRES methods. We believe that the required number of iterations would grow much faster if finer meshes are used than that of the Schwarz preconditioned GMRES methods.

4.4. A comparison with a global ILU(0) preconditioned GMRES method. In Table 5, we show the maximum and total number of iterations when using the global ILU(0) preconditioned GMRES method for both test cases. In terms of the number of EMatVec, the method outperforms, by a factor of 5% to 30%(Comparing Table 5 and the bottom part of Table 2), the multiplicative VDS preconditioned GMRES methods we discussed in the paper. Unfortunately, its parallelization on distributed memory computers is not very easy.

4.5. The effect of frozen preconditioner. Finally, we examined the effect of using the same preconditioner, or part of the preconditioner, for several time steps without doing the factorization at every time step. In Table 6, we summerize the results for using different numbers of frozen steps, namely Froz $= 1, 5, ...$ There is a range of optimal Froz one can choose from; similar numbers of EMatVec were obtained in our implementation for Froz ranging from 5 to 50. For the subsonic case, we can go a bit further, e.g., take Froz = 100.

5. Conclusions. We proposed and tested a family of variable degree Schwarz (VDS) preconditioned GMRES methods for solving linear systems that arise from the discretization of unsteady, compressible N.-S equations on 2D unstructured meshes for both subsonic and transonic flows past a single element NACA0012 airfoil. We found that with implicit methods, larger time steps can be used and the overall simulation time can be reduced significantly comparing with the explicit method.

In VDS, the level of preconditioning in each subdomain varies according to the local flow condition, therefore extra preconditioning is performed only when and where it is needed. For subsonic problems, we found that the conditioning of the subdomain matrices changes quite a bit from one flow region to another, and extra local preconditioning in subdomains in which the flow changes drastically can significantly reduce the total number of global linear iterations. This is somewhat less obvious for transonic flow, which needs a nearly uniformly small global and local number of iterations.

When using VDS, the best results are obtained with small overlap. For the multiplicative version, the convergence rate depends very mildly on the number of subdomains (up to 128 subdomains has been tested), and for the additive version, a slight dependence is observed for the subsonic test problem and therefore a coarse space might be useful. We also compared our methods with the simple point(means 4×4 block for our multicomponent problems) Jacobi iterative method and the global ILU(0) preconditioned GMRES method. We found that Jacobi is significantly slower than the proposed methods, especially for the subsonic case, and if sequential computers are the primary computing platform, then the global ILU(0) preconditioned GMRES is a winner over all methods we have tested.

All of our tests were done on a sequential computer. Considerable effort is needed in order to obtain a well-balanced parallel implementation. We remark that our mesh partitioning is obtained without the knowledge of the flow condition and our experiences show that a solution dependent mesh partitioning would offer a more computationally balanced decomposition of the problems.

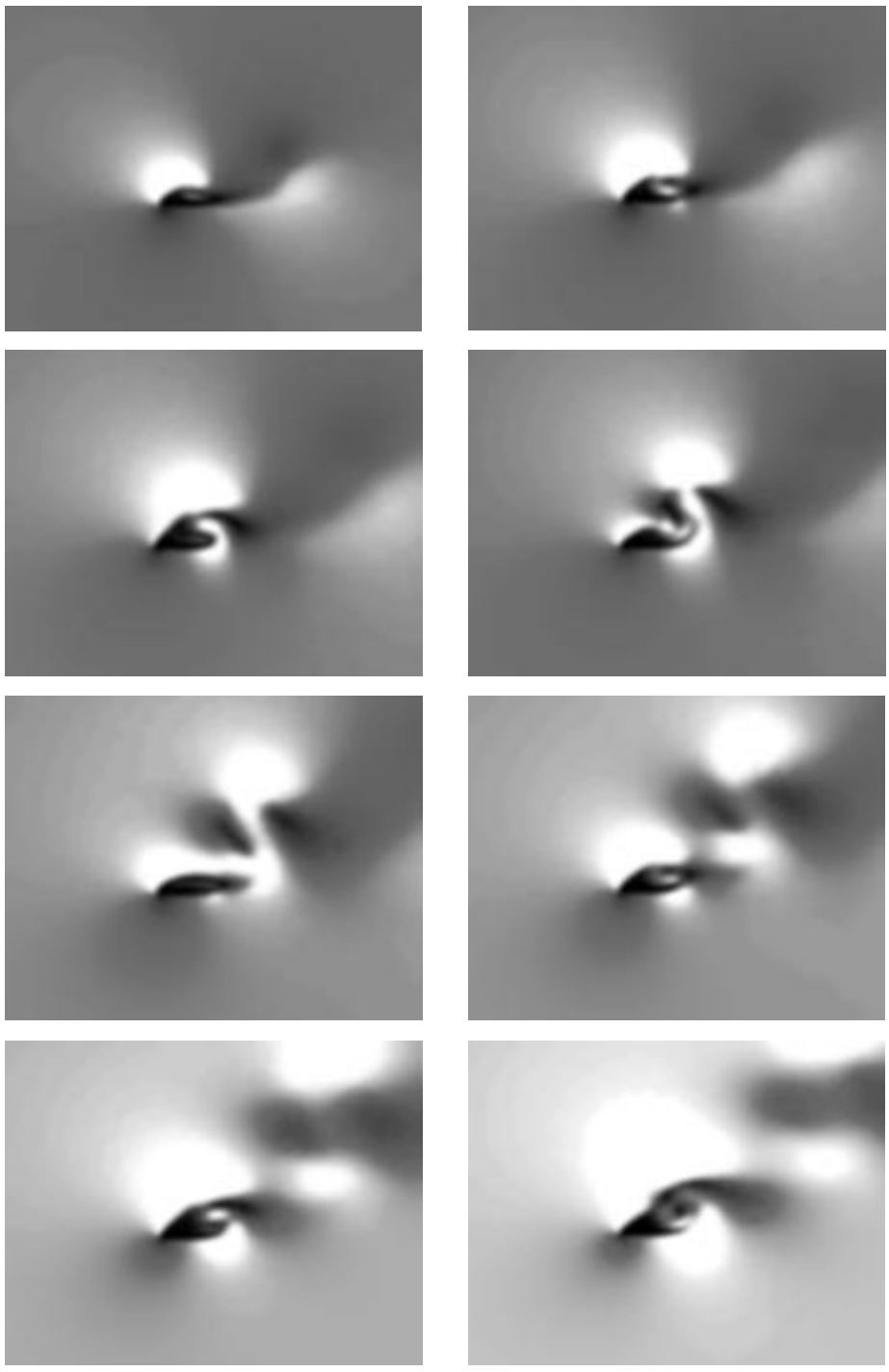

FIG. 5. *The Mach distribution at* $M_\infty = 0.1$*, for non-dimensionalized time* $t = 2, 3, ..., 9$.

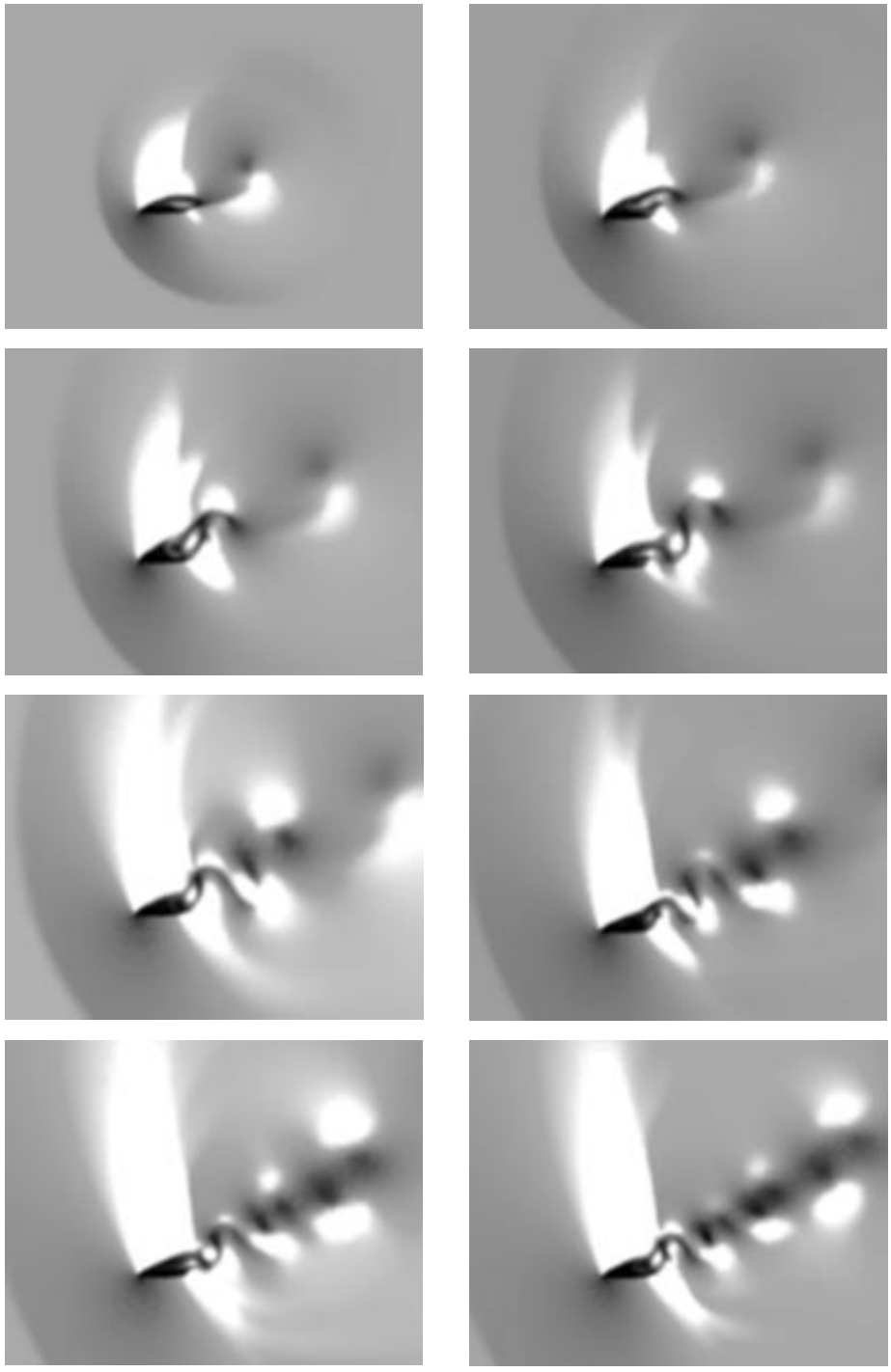

FIG. 6. *The Mach distribution at* $M_\infty = 0.84$*, for non-dimensionalized time* $t = 2, 3, ..., 9$.

REFERENCES

[1] T. J. Barth, T. Chan, and W. P. Tang, *Implicit parallel preconditioning techniques for computational fluid dynamics*, in Proceedings of the Copper Mountain Conference on Iterative Methods, T. Manteuffel and S. McCormick, eds., Copper Mountain, Colorado, 1996.

[2] T. J. Barth and S. W. Linton, *An unstructured mesh Newton solver for compressible fluid flow and its parallel implementation*, AIAA Paper 95-0221, (Jan. 1995).

[3] R. M. Bean and R. F. Warming, *An implicit finite-difference algorithm for hyperbolic systems in conservation laws*, J. Comp. Phys., 22 (1976), pp. 87–110.

[4] X.-C. Cai, C. Farhat, and M. Sarkis, *Schwarz methods for the unsteady compressible Navier-Stokes equations on unstructured meshes*, in Domain Decomposition Methods in Sciences and Engineering, R. Glowinski, J. Periaux, Z. Shi, and O. Widlund, eds., England, 1996, John Wiley & Sons, Ltd. To appear.

[5] X.-C. Cai, W. D. Gropp, D. E. Keyes, R. G. Melvin, and D. P. Young, *Parallel Newton-Krylov-Schwarz algorithms for the transonic full potential equation*, SIAM J. Sci. Comput., (1996). Submitted.

[6] X.-C. Cai and Y. Saad, *Overlapping domain decomposition algorithms for general sparse matrices*, Numer. Lin. Alg. Applics, 3 (1996), pp. 1–18.

[7] X.-C. Cai and M. Sarkis, *Local multiplicative Schwarz algorithms for convection-diffusion equations*, Tech. Rep. ICASE Report No. 95-86, ICASE, NASA Langley Research Center, 1995.

[8] X.-C. Cai and O. Widlund, *Multiplicative Schwarz algorithms for nonsymmetric and indefinite elliptic problems*, SIAM J. Numer. Anal., 30 (1993), pp. 936–952.

[9] T. F. Chan and T. P. Mathew, *Domain decomposition algorithms*, Acta Numerica, (1994).

[10] Digital Equipment Corporation, *Digital Extended Math Library for DEC OSF/1 AXP: Reference Manual*, Digital Equipment Corporation, 1993.

[11] M. Dryja and O. B. Widlund, *Domain decomposition algorithms with small overlap*, SIAM J. Sci. Comp., 15 (1994), pp. 604–620.

[12] C. Farhat, L. Fezoui, and S. Lanteri, *Two-dimensional viscous flow computation on the Connection Machine: Unstructured meshes, upwind schemes and parallel computation*, Comput. Methods Appl. Mech. Engrg., 102 (1993), pp. 61–88.

[13] C. Farhat and S. Lanteri, *Simulation of compressible viscous flows on a variety of MPPs: Computational algorithms for unstructured dynamic meshes and performances results*, Comput. Methods Appl. Mech. Engrg., 119 (1994), pp. 35–60.

[14] C. Farhat, S. Lanteri, and H. Simon, *TOP/DOMDEC: A software tool for mesh partitioning and parallel processing and applications to CSM and CFD computations*, Comput. Sys. Engrg., 6 (1995), pp. 13–26.

[15] L. Fezoui and B. Stoufflet, *A class of implicit upwind schemes for Euler simulations with unstructured meshes*, J. Comp. Phys., 84 (1989), pp. 174–206.

[16] P. Forsyth and H. Jiang, *Iterative methods for full Newton Jacobian for compressible Navier-Stokes equations*, in Proceedings of the Copper Mountain Conference on Iterative Methods, T. Manteuffel and S. McCormick, eds., Copper Mountain, Colorado, 1996.

[17] G. Golub and J. M. Ortega, *Scientific Computing: An Introduction with Parallel Computing*, Academic Press, Inc., 1993.

[18] W. D. Gropp, D. E. Keyes, and J. S. Mounts, *Implicit domain decomposition algorithms for steady, compressible aerodynamics*, in Sixth Conference on Domain Decomposition Methods for Partial Differential Equations, A. Quarteroni, J. Periaux, Y. A. Kuznetsov, and O. B. Widlund, eds., Providence, RI, 1994, AMS.

[19] W. D. Gropp, B. F. Smith, and L. C. McInnes, *PETSc 2.0 User's Manual*, Tech. Rep. ANL-95/11, Argonne National Laboratory, 1995.

[20] C. Hirsch, *Numerical Computation of Internal and External Flows, Vol I*, Wiley, New York, 1990.

[21] D. Knoll, *Newton-Krylov-Schwarz methods applied to the Tokamak edge plasma fluid equations*, in Domain-Based Parallelism and Problem Decomposition Methods in Computational Science and Engineering, D. Keyes, Y. Saad, and D. Truhlar, eds., Philadelphia, Pennsylvania, 1994, SIAM.

[22] A. Pothen, H. D. Simon, and K.-P. Liou, *Partitioning sparse matrices with eigenvectors of graphs*, SIAM J. Matrix Anal. Appl., 11 (1990), pp. 430–452.

[23] T. H. Pulliam, *Low Reynolds number numerical solutions of chaotic flows*, AIAA Paper 89-0123, (1989). 27th Aerospace Sciences Meeting, Reno, Nevada.

[24] P. R. Roe, *Approximate Riemann solvers, parameters vectors and differences schemes*, J. Comp. Phys., 43 (1981), pp. 357–371.

[25] Y. Saad, *A flexible inner-outer preconditioned GMRES algorithm*, SIAM J. Sci. Stat. Comput., 14 (1993), pp. 461–469.

[26] Y. Saad and M. H. Schultz, *GMRES: A generalized minimum residual algorithm for solving non-symmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869.

[27] B. F. Smith, P. E. Bjørstad, and W. D. Gropp, *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press, 1996.

[28] J. Steger and R. F. Warming, *Flux vector splitting for inviscid gas dynamic with application to finite-difference methods*, J. Comp. Phys., 40 (1981), pp. 263–293.

[29] B. van Leer, *Towards the ultimate conservation difference schemes V: A second-order sequel to Goudonov's method*, J. Comp. Phys., 32 (1979), pp. 361–370.

[30] V. Venkatakrishnan, *A perspective on unstructured grid flow solvers*, Tech. Rep. ICASE Report No. 95-3, ICASE, NASA Langley Research Center, Feb. 1995.

[31] V. Venkatakrishnan and D. J. Mavriplis, *Implicit method for the computation of unsteady flows on unstructured grids*, Tech. Rep. ICASE Report No. 95-60, ICASE, NASA Langley Research Center, Aug. 1995.

TABLE 2

Each test is for 100 time steps and at each time step the initial preconditioned residual is reduced by a factor of $\tau = 10^{-3}$ by using GMRES/VDS with ovlp = 1. We use GMRES/ILU(0) as inexact local solvers to reduce the local preconditioned residual by a factor of $\delta = 10^{-1}$. Here CFL=50.

ASM	**Test 1**			**Test 2**		
# subdomains	MaxIt	TotalIt	EMatVec	MaxIt	TotalIt	EMatVec
8	6	545	3150	6	519	1471
16	7	585	3529	6	506	1628
32	9	673	3851	6	560	1864
64	10	756	4223	6	600	2184
128	11	842	5621	7	603	2192
MSM	**Test 1**			**Test 2**		
8	4	292	1613	3	300	832
16	4	316	1812	3	300	915
32	4	320	1834	3	300	994
64	4	344	1900	3	300	1089
128	4	351	2335	3	300	1084

TABLE 3

GMRES iteration numbers to reduce the preconditioned residual of **Test 1** *to $\tau = 10^{-6}$ using GMRES/(additive VDS) with 8 subdomains. We use GMRES/ILU(0) as inexact local solver with different local stopping criteria. Here CFL=100.*

	ovlp = 0	ovlp = 1	ovlp = 2	ovlp = 3
1 iteration	35	43	46	48
$\delta = 5.0e^{-1}$	23	32	33	33
$\delta = 3.0e^{-1}$	18	20	19	19
$\delta = 1.0e^{-1}$	16	15	14	14
$\delta = 1.0e^{-2}$	15	12	11	10
exact	15	11	10	9

TABLE 4

The maximum (MaxIt) and total (TotalIt) local GMRES/ILU(0) iteration numbers. The global solver is GMRES/(multiplicative VDS). The parameters are $\tau = 10^{-3}$, $\delta = 10^{-1}$, ovlp = 1 and the local solvers are $ILU(0)$. Here CFL=50.

	Ω_1	Ω_2	Ω_3	Ω_4	Ω_5	Ω_6	Ω_7	Ω_8
Test 1, MaxIt	2	2	2	3	4	6	3	5
TotalIt	150	106	113	225	307	597	291	421
Test 2, MaxIt	2	2	2	2	2	2	1	1
TotalIt	127	200	109	107	132	185	150	100

TABLE 5

The number of equivalent EMatVec operations needed for 100 time steps starting at $t = 1.0$. $\tau = 10^{-3}$, $\delta = 10^{-1}$, CFL=50.

	4×4 Jacobi	Global ILU(0)
Test 1, MaxIt	74	10
EMatVec	7299	1472
Test 2, MaxIt	46	4
EMatVec	4222	800

TABLE 6

The number of EMatVec operations needed for 100 time steps starting at $t = 1.0$. In GMRES/(multiplicative VDS), $\tau = 10^{-3}$, $\delta = 10^{-1}$, CFL=50, number of subdomains is 8 and ovlp = 1. For the Froz=200 case, the numbers are taken for 200 time steps divided by 2.

Froz=	1	5	10	50	100	200
Test 1, EMatVec	1613	1611	1615	1618	1629	1875
TotalIt	292	292	291	290	291	321
Test 2, EMatVec	832	829	830	842	868	1243
TotalIt	300	300	300	305	315	469